Software Product Lines
A New Paradigm for the New Century

Paul Clements
Software Engineering Institute

Software developed as a product line promises to be a dominant development paradigm for the new century, one that the Department of Defense (DoD) can leverage when acquiring software-intensive systems. This article discusses the advantages of product lines, uncovers some of their pitfalls, and shows by example the kinds of successes the organizations can enjoy.

Imagine turning out a 1.5 million-line Ada command and control system for a Navy frigate warship. The system is hard real-time, fault-tolerant, and highly distributed, running on 70 separate processors on 30 different local area network nodes scattered all over the ship. It must interface with radars and other sensors, missile and torpedo launchers, and other complicated devices. The human-computer interface is complex and highly demanding. In this application, quality is everything: The system must be robust, reliable, and avoid a host of performance, distribution, communication, and other errors.

Now suppose that you have not one of these systems to build but several. Your marketing department has succeeded beyond your wildest dreams. Navies from all over the world have ordered your command and control system. Now, your software must run on almost a dozen different ship classes including a submarine, systems that are drastically separate: The end users speak different languages (therefore, the human-computer interface requirements are extremely different), the ships are laid out differently, have different numbers of processors and nodes, and different fault tolerance requirements, different weapons systems and sensors, and different computers and operating systems. But quality remains crucial in all of them.

Suppose you are the manager for this megaproject. Do you panic? Do you resign? Run to a third-world country? What if you could produce each one of the systems for a fraction of the cost and in a fraction of the time that one would normally expect? And what if you could do it so that quality was improved and reliability and customer satisfaction increased with each new system? What if creating a new ship system was merely a matter of combining large, easily tailorable components under the auspices of a software architecture that was generic across the entire domain (in this case, of shipboard command and control systems)?

Is this a fantasy? No, it is not. It is the story of CelsiusTech Systems AB, a long-time European defense contractor. In the 1980s, CelsiusTech was confronted with the dilemma outlined above: They had to build two large command and control systems, each larger than anything they had attempted before, and they had barely enough resources to build one. Because necessity stimulates invention (and determination implements it), CelsiusTech realized that their only hope was to build both systems at once *using the same assets and resources*. And in a visionary stroke, CelsiusTech knew that their future lay in exploiting these assets for not only the first two systems but also for a whole family of products they hoped and expected would follow.

Software Product Lines

In short, CelsiusTech launched a *software product line*. A product line is a set of products that together address a particular market segment or fulfill a particular mission. Product lines promise to become the dominating production software paradigm of the new century. Product flexibility is the new anthem of the marketplace, and product lines fulfill the promise of tailor-made systems built specifically for the needs of particular customers or customer groups. What makes product lines succeed from the vendor's (developer's) point of view is that the commonalities shared by the products can be exploited to achieve economics of production.

Product lines are nothing new in manufacturing. Boeing builds one, so does Ford, IBM, and even McDonald's. Each of these exploits commonality in different ways. Boeing, for example, developed the 757 and 767 transports in tandem, and the parts lists of these two decidedly different aircraft overlap by about 60 percent. But *software* product lines based on interproduct commonality are a relatively new concept, and the community is discovering that this path to success contains more than its share of pitfalls.

The Software Engineering Institute (SEI) has a program to identify and promulgate the best practices for product-line production and help organizations negotiate the hurdles to which adopting a product-line approach will lead. The Product-Line Systems Program focuses on these essential technology areas for product-line production:

- Domain Engineering – Reveals the commonalities and variations among a set of products.
- Architecture – The foundation for a product line, it provides the framework into which tailorable components plug.
- Architecture-Based Development – The disciplined derivation or generation of product components (and once the components are ready, whole products) from the architectural skeleton.
- Reengineering – helps mine reusable assets from legacy assets.

The result is a technology infrastructure that can produce large custom systems quickly and reliably by checking out components from the asset repository, tailoring the components for their particular application (CelsiusTech uses compile-time parameters to instantiate different versions of a component), and beginning the integrate-and-test cycle as in normal system development.

Product Line Benefits

Once the product-line repository is established, consider what is saved each time a product is ordered.

- Requirements. Most of the requirements are common with earlier systems and therefore can be used. Requirements analysis is saved. Feasibility is assured.
- Architectural design. An architecture for a software system represents a large investment in time from the organization's most talented engineers. The quality goals for a system—performance, reliability, modifiability, etc.—are largely allowed or precluded once the architecture is in place. If the architecture is wrong, the system cannot be saved; however, for a new product, this most important design step is already done and need not be repeated.
- Components. The detailed (internal) designs for the architectural components are reused from system to system, as is the documentation of those designs. Data structures and algorithms are saved and need not be reinvented.
- Modeling and analysis. CelsiusTech reports that the real-time distributed headache associated with the kinds of systems they build (real-time distributed) has all but vanished. When they field a new product in their product line, they have extremely high confidence that the timing problems have been worked out, and the challenges associated with distributed computing—synchronization, network loading, and absence of deadlock—have been eliminated.
- Testing. Test plans, test processes, test cases, test data, test harnesses, and the communication paths required to report and fix problems are already available.
- Planning. Budgets and schedules can be reused from previous projects, and they are much more reliable.
- Processes. Configuration control boards, configuration management tools and procedures, management processes, and the overall software development process are in place, have been used before, and are robust, reliable, and responsive to the organization's special needs.
- People. Because of the commonality of the applications, personnel can be fluidly transferred among projects as required. Their expertise is applicable across the entire line.

Product lines enhance quality. Each new system takes advantage of all of the defect elimination in its forebearers; both developer and customer confidence rise with each new instantiation. The more complicated the system, the higher the payoff for solving the vexing performance, distribution, reliability, and other engineering issues only once for the entire family.

Clearly, product lines benefit the developing organization, but they also benefit acquirers of systems as well. Acquiring a family of related systems using a product-line acquisition approach (as opposed to acquiring each system separately and independently) clearly falls within the realm of DoD reuse initiatives and policies and promises to accrue significant benefits for the DoD, including

- Streamlining the acquisition process.
- Enjoying higher product quality.
- Lower acquisition cost.
- Simplified training.
- Reduced maintenance cost.

Organizational Maturity Needs

It takes a certain maturity in the developing organization to successfully field a product line. Technology is not the only barrier to successful product-line adoption. Experiences in the Product-Line Systems Program show that organization, process, and business issues are equally vital to master.

For instance, traditional organizational structures that have one business unit per product are generally not appropriate for product lines. Who will build and maintain the core reusable assets—the architecture, the reusable components, and so forth? If these assets are under the control of a business unit associated with one product or one large customer, the assets may evolve to serve that business unit, that product, and that customer to the exclusion of the others. On the other hand, to establish a separate business unit to work on the core assets but be divorced from working on individual products carries the danger that this unit will produce assets that emphasize beauty and elegance over practicality and utility. In either case, producing and managing the reusable assets means establishing processes to make the assets satisfy the needs of all of the business units that use them. This is a crucial role that requires staff skilled in abstraction, design, negotiation, and creative problem solving. The question of funding the core asset development is crucial.

Customer Management

Customer management becomes an important product-line function. Customers interact with a product-line organization in a different way. Marketers can no longer agree to anything customers want but must instead nudge customers to set their requirements so that they can be fulfilled by a version of the product line within the planned scope of variation.

Contrary to intuition, this often makes the customer much happier than before. Under the new paradigm, the marketer can point to specific requirements that would put the customer's new system outside the scope of the product line, which would increase the cost and delivery time, lower the system's reliability, and keep that customer out of a community of customers to which the vendor pays a lot of attention. Thus, the customer could clearly (and probably for the first time) see the real cost of those "special" requirements and make an informed decision about their real value. If the customer decides

that a variant of the "standard" or product-line system will suffice, so much the better. If not, the customer can still order a system to satisfy particular requirements but with a better idea of where the risks may be hiding.

The customer community should not be underestimated. In CelsiusTech's case, their naval customers around the world banded together to form a users' group. They did this in their self-interest—to provide a forum in which they could jointly derive new requirements for their evolving systems and drive CelsiusTech to supply new systems more economically than they otherwise might. But it does not take much to realize how beneficial this is to CelsiusTech as well: Their customer base is jointly defining their next-generation products and effectively buying in to their approach, thus guaranteeing the vitality of their product line for years to come.

The users' group has a clear lesson for DoD acquisitions: It pays to collaborate (or at least communicate) when it comes to commissioning or purchasing similar systems.

Conclusion

Successful transition to product-line technology is thus a careful blend of technology, process, organization, and business factors improvement. The Product-Line Systems Program is attempting to codify these practices and understand how they vary with the type of organization involved and the kind of systems being built. Through a series of workshops, case studies, and collaborative engagements, SEI is helping to build a community of organizations interested in moving to a product-line approach for their software products.

We believe that product lines will be the predominant software paradigm at the beginning of the new century. The history of programming can be viewed as an upward spiral in which the abstractions manifested by components have grown larger and more application meaningful, with resulting increases in the reuse and applicability of those components. From subroutines in the 1960s to modules in the 1970s to objects in the 1980s to component-based systems in the 1990s, software product lines will perpetuate the upward spiral by accomplishing previously unheard-of levels of reuse from system to system.

If the pitfalls are successfully negotiated, the result is an enviable capacity to deliver extremely large systems on time and within budget.

For more information about the Product-Line Systems Program and its technology initiatives, visit SEI's Web page at http://www.sei.cmu.edu. You can download the full report that details the CelsiusTech product-line case study, which includes data about their dramatic results in time to market, levels of reuse, and required staffing. You also can read other product-line-related material, including the latest version of the SEI's Product-Line Practice Framework, a document that describes the essential practice areas of successful product-line development and acquisition. Contact the program manager, Linda Northrop, at lmn@sei.cmu.edu, for additional information. ◆

About the Author

Paul Clements is a senior member of the technical staff at Carnegie Mellon University's SEI. A graduate of the University of North Carolina and the University of Texas, he is a project leader in the SEI's Product-Line Systems Program. His work includes collaborating with organizations that are launching product-line efforts. He is a co-creator of the Software Architecture Analysis Method (SAAM), which allows organizations to evaluate architectures for fitness of purpose. He and others are working on an extension to SAAM, which will allow analysis of quality attribute trade-offs at the architectural level. He is co-author of *Software Architecture in Practice* (Addison-Wesley-Longman, 1998) and over three dozen papers and articles about software engineering.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Voice: 512-453-1471
Fax: 412-268-5758
E-mail: clements@sei.cmu.edu